INFORMATIK

Universität des Saarlandes
FR Informatik

Harald Ganzinger
Andreas Podelski
Uwe Waldmann

9. April 2001

Übungen zur Vorlesung „Programmiersprachen"
Übungsblatt 1

Programmieraufgabe 1.1: (*4 P.*)
Schreiben Sie eine ML-Funktion `primes`, die eine natürliche Zahl n als Argument bekommt und die Liste aller Primzahlen kleiner oder gleich n in aufsteigender Reihenfolge liefert. Nutzen Sie dazu die folgende Charakterisierung von Primzahlen aus: Eine natürliche Zahl $m > 1$ ist genau dann eine Primzahl, wenn $m \bmod p > 0$ für alle Primzahlen p mit $p^2 \leq m$ gilt. (Vorsicht: nicht mehr als n rekursive Aufrufe von `primes` verwenden!)

Programmieraufgabe 1.2: (*2 P.*)
Schreiben Sie eine ML-Funktion `fromfirstcomm` mit den folgenden Eigenschaften:

- `fromfirstcomm` bekommt zwei Listen l_1 und l_2 als Argumente.
- Falls $l_1 = [x_1, x_2, \ldots, x_n]$ und x_i das erste Element von l_1 ist, das auch in l_2 vorkommt, dann liefert `fromfirstcomm` die Liste $[x_i, \ldots, x_n]$.
- Falls l_1 und l_2 kein gemeinsames Element besitzen, dann liefert `fromfirstcomm` die leere Liste.

Programmieraufgabe 1.3: (*4 P.*)
Schreiben Sie eine ML-Funktion `distrib`, die Studenten auf Übungsgruppen verteilt. Das erste Argument von `distrib` ist eine Liste von Paaren der Form (s, stl); dabei ist der String s der Name eines Studenten, und stl ist die Liste derjenigen Termine,[1] an denen s an einer Übungsgruppe teilnehmen könnte (nach Präferenz geordnet, der bevorzugte Termin zuerst). Das zweite Argument von `distrib` ist die Liste g derjenigen Termine, an denen tatsächlich Übungsgruppen stattfinden. Das Ergebnis von `distrib` ist eine Liste von Paaren der Form (t, tsl); dabei ist t ein Termin aus g und tsl die Liste aller Studenten, die t unter den Terminen aus g favorisieren. Termine aus g, denen kein Student zugeordnet ist, und Studenten, die an keinem Termin aus g Zeit haben, tauchen im Ergebnis nicht auf. (Sie können die Funktion `fromfirstcomm` aus der vorherigen Aufgabe verwenden.)

[1] Termine können beispielsweise durch ganze Zahlen bezeichnet werden.

Beispiel:

```
distrib  [("Udo Weber",  [4,5,1,2,3]),
          ("Eva Meyer",  [1,3,5,2]),
          ("Kai Lange",  [5,4]),
          ("Julia Alt",  [4,3,2,1,5])]  [1,2,3]
```

liefert das Ergebnis `[(3,["Julia Alt"]),(1,["Udo Weber","Eva Meyer"])]`.

Aufgabe 1.4: (*3 P.*)
Erklären Sie, warum das ML-System unten beim Aufruf `f 3` den Wert 6 und nicht etwa den Wert 15 ausgibt.

```
- val x = 2;
val x = 2 : int
- fun f y = x * y;
val f = fn : int -> int
- val x = 5;
val x = 5 : int
- f 3;
val it = 6 : int
```

Aufgabe 1.5: (*3 P.*)
Unterscheiden sich die Deklarationen

```
fun g x y = 2 * x + y;
fun g(x,y) = 2 * x + y;
fun g(x)(y) = 2 * x + y;
```

voneinander? Wenn ja: wie?

Aufgabe 1.6: (*4 P.*)
Die Eingabe `["a",1];` führt zu folgender Fehlermeldung:

```
stdIn:33.1-33.8 Error: operator and operand don't agree [literal]
  operator domain: string * string list
  operand:         string * int list
  in expression:
    "a" :: 1 :: nil
```

Erklären Sie, was die einzelnen Teile dieser Fehlermeldung bedeuten.

Schicken Sie die ausgetesteten Lösungen der Programmieraufgaben (inklusive einiger Testbeispiele mit Ergebnissen) bis zum 17. 4., 13:00 Uhr per Mail an Ihren Übungsgruppenleiter, Subject: `PS Blatt 1`. Bereiten Sie die übrigen Aufgaben so vor, daß Sie die Lösung in der nächsten darauffolgenden Übungsstunde vortragen können.